# 通過テスト６ 解答

## 6 参照とオブジェクト

1. 問１

＜Box クラス＞

```
public class Box {
     private int size;
     private double weight;

     public Box(int size, double weight){
          this.size=    size;
          this.weight   =   weight;
     }
     public int getSize(){
          return    size;
     }
     public double getWeight(){
          return    weight;
     }
}
```

【解説】カプセル化に配慮したクラスとなっている.

＜Carrage クラス＞

```
public class Carrage {
    public int total;
    public void add(Box b){
        total += calc(b) + 200; // 200 円は基本料金;
    }
    int calc(Box b){
        double  w   =   b.getWeight();
        int     s   =   b.getSize();
        if(w<=2){
            if(s<=100){
                return  500;
            }else if(s<=300){
                return  1000;
            }else{
                return  2000;
            }
        }else if(w<=5){
            if(s<=100){
                return  700;
            }else if(s<=300){
                return  1200;
            }else{
                return  2200;
            }
        }else{
            if(s<=100){
                return  1000;
            }else if(s<=300){
                return  1500;
            }else{
                return  3000;
            }
        }
    }
    public int getTotal(){
        return  total;
    }
}
```

【解説】add メソッド, calc メソッドの引数が Box 型の参照であるところがポイントである. 引数の b から重さとサイズを得て計算処理を行う. オブジェクト指向の最も一般的なプログラムである.

```
public class Exec1 {
     public static void main(String[] args) {
          int[]    s    =   {70, 150, 270, 90, 310, 180, 120};
          double[] w    =   {3.2, 2.8, 7.1, 3.0, 4.5, 3.7, 1.5};

          Carrage  c    =   new Carrage();
          for(int i=0; i<s.length; i++){
               Box      b   =   new      Box(s[i], w[i]);
               c.add(b);
          }
          System.out.println("合計金額="+c.getTotal());
     }

}
```

【解説】このように配列を使うと処理が簡単になる．for 文の中で配列データを使って Box 型オブジェクトを作成し，これを使って，Carrage クラスの add メソッドで運賃を計算する．最後に getTotal メソッドで合計を表示する． 実行すると 合計金額=9900 と表示される．

2．問 1

```
public class Point {
     private double x;
     private double y;

     public Point(double x, double y){
          this.x = x;
          this.y = y;
     }
     public Point(double a){
          this(a, a);
     }
     public double getX(){ return      x; }
     public double getY(){ return      y; }
     public Point  add(Point p){return  new     Point(x+p.x, y+p.y);}
     public Point  sub(Point p){return  new     Point(x-p.x, y-p.y);}
     public Point  mul(Point p){return  new     Point(x*p.x, y*p.y);}
     public Point  div(Point p){return  new     Point(x/p.x, y/p.y);}
     public String toString(){return   "("+ x + ", " + y + ")"; }
}
```

【解説】面倒だが，ひとつずつのメソッドを指示通りに作成するだけでよい．return 文の中で新しくオブジェクトを作成するように書くと，簡単になる．

問 2

```
public class Exec2 {

     public static void main(String[] args) {

          Point    p_2  =   new Point(2);
          Point    p1   =   new Point(10.5, 12.4);
          Point    p2   =   new Point(2.4, 10.7);
          Point    p3   =   new Point(0.3, -0.5);

          Point    p0   =   new Point(0, 0);
          p0            =   (p_2.mul(p3).mul( p1.add(p2) )).sub(p3);
          System.out.println("ans=" + p0);

     }

}
```

【解説】演算結果として Point 型のオブジェクトが返されることに注意する.

```
(p_2.mul(p3).mul( p1.add(p2) )).sub(p3)
```

のように次々にメンバ参照演算子を使って連結できる.

3. 問 1

```
public class Calc {
     private Point[] p;
     public Calc(Point[] p){
          this.p    =    p;
     }
     public int size(){ return  p.length; }
     public Point total(){
          Point     p0   =   new Point(0);
          for(int i=0; i<p.length; i++){
               p0 = p0.add(p[i]);
          }
          return    p0;
     }
     public Point average(){
          Point p0 = total();
          return    p0.div(new Point(p.length));
     }
}
```

【解説】オブジェクトの配列をフィールドに持つ.コンストラクタ引数が配列の参照になる点こ
とに注意する．オブジェクト指向のプログラムではよくあるケースである.

問 2

```
public class Exec3 {
     public static void main(String[] args) {

          Point[] p = {new Point(1.5, 6.5), new Point(2.8, -1.2),
                       new Point(-3.8, 4.3), new Point(2.4, 3),
                       new Point(1.8, 2.1), new Point(-2.2, -2.8),
                       new Point(4.1, 5.7)
                      };

          Calc c    = new Calc(p);
          System.out.println("合計＝" + c.total());
          System.out.println("平均＝" + c.average());

     }

}
```

【解説】配列の要素を new 文で作成して列挙する方法に慣れること.